Un Algorithme de Colonie de Fourmis pour la Planification de Formations en Santé *

Simon Caillard, Corinne Lucet, Laure Brisoux-Devendeville

Laboratoire MIS (UR 4290), Université de Picardie Jules Verne
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
{simon.caillard, corinne.lucet, laure.devendeville}@u-picardie.fr

Mots-clés : *recherche opérationnelle, optimisation, colonie de fourmis, tabou, santé, emploi du temps*

1 Introduction

Le centre SimUSanté, situé à Amiens, en France, propose plus de 400 formations dans les différents domaines de la santé. Ces formations sont à destination de nombreux publics différents et proposent un apprentissage au travers de diverses simulations et cas pratique. Afin de pouvoir réaliser ces formations, le personnel du centre (formateurs et techniciens) dispose de multiples compétences, et les salles sont variées et modulables. Réaliser un emploi du temps des sessions de formations, qui respecte l'ensemble des contraintes (de temps et de ressources) est un objectif difficile, et de première importance pour SimUSanté. Le problème du centre est dérivé de celui du *Curriculum-Based Course Timetabling problem* (CB-CTT)[4] qui est NP-Difficile[3]. Afin de résoudre ce problème, nous proposons un algorithme de Colonie de Fourmis, associé à un système tabou qui empêche les fourmis d'être piégées dans des minimums locaux.

2 Définition du problème

L'horizon T est un ensemble de créneaux sur lesquels toutes les activités de A doivent être planifiées, en leur attribuant des ressources, employés et salles de l'ensemble R. Les ressources ne sont pas toujours disponibles sur tout l'horizon. Ainsi $isavailable_r^t = 1$, si la ressource $r \in R$ est disponible au créneau $t \in T$, 0 sinon. De plus, chaque ressource $r \in R$ est associée à un ou plusieurs types de ressources. Un type de ressource représente une compétence ou une caractéristique de salle. On note Λ l'ensemble de ces types. On cherche alors à planifier un ensemble S de sessions de formations. Chaque session $s \in S$ est composée d'un ensemble d'activités A_s, ou toute activité $a \in A_s$ a une durée $duration_a$, un ensemble d'activités qui doivent la précéder $pred_a$ et une quantité $qtreq_\lambda^a$ de ressources de type λ, $\forall \lambda \in \Lambda$, $qtreq_\lambda^a$, nécessaire à sa réalisation.

Le problème de SimUSanté peut être modélisé sous la forme d'un graphe $\mathcal{G} = ((A, T, R), E)$, où A, T et R sont respectivement les ensembles des activités, des créneaux et des ressources. $E = \{A \times T\} \cup \{(t, r) | t \in T, r \in R \text{ et } isavailable_r^t = 1\} \cup \{R \times R\}$. Un chemin (a, t, c) représente une planification possible de l'activité $a \in A$, sur l'intervalle $[t; t + duration_a[$, $t \in T$, avec l'utilisation des ressources $c = \{r_1, r_2, \ldots, r_k\} \subseteq R$, qui répondent strictement à ses besoins. Une solution Sol est alors une collection de chemins $\{(a_1, t_1, c_1), \ldots, (a_n, t_n, c_n)\}$, avec $n \leq |A|$. On note UA l'ensemble des activités qui n'ont pu être planifiées, et Sol_s l'ensemble des chemins relatifs à la session $s \in S$.

Notre objectif est de planifier des activités, tout en minimisant la fonction objectif $Eval()$ définie par l'équation 1. $Eval(Sol)$ calcule pour une solution Sol la somme des makespans

*Ce projet est supporté par la région Hauts de France et le centre SimUSanté/CHU Amiens-Picardie

sur l'ensemble des sessions, plus la somme des pénalités associées aux activités non planifiées. Minimiser $Eval()$ revient donc à planifier un maximum d'activités, tout en compactant l'emploi du temps.

$$Eval(Sol) = \sum_{s \in S} (\max_{(a,t,c) \in Sol_s} t + duration_a - \min_{(a,t,c) \in Sol_s} t) + |UA| \times |T| \quad (1)$$

3 Résolution

SimUTACO, un algorithme basé sur l'algorithme *Ant Colony System* [5], est utilisé pour résoudre le problème de SimUSanté. Ce type d'algorithme s'inspire du comportement des fourmis qui déposent des phéromones sur leur chemin afin de guider d'autres fourmis et permettre ainsi de trouver le chemin le plus rapide entre leur nid et une source de nourriture. *SimUTACO* tend à éviter la convergence trop rapide des algorithmes ACO, par sa gestion des phéromones (alternance d'exploration et d'exploitation), mais aussi par l'utilisation d'un système tabou empêchant les fourmis de rester dans une même zone de l'espace des solutions.

À chaque itération i, chaque fourmi k construit une solution $Sol^i_k = \{(a_1, t_1, c_1), \ldots, (a_n, t_n, c_n)\}$ qui représente l'ensemble des chemins qu'elle a emprunté. Puis, les phéromones correspondant aux chemins de la pire solution $Sol^i_{worst} = \text{argmax}_{k \in K}(Eval(Sol^i_k))$, construite à l'itération i sont évaporées de $\gamma\%$. Des phéromones additionnelles sont alors déposées sur les chemins des solutions Sol^i_{best} et Sol_{best}, qui correspondent à la meilleure solution trouvée respectivement à l'itération i, et depuis le début de l'algorithme. Enfin, si à une itération i, le pourcentage de différence entre les quantités de phéromones des chemins de Sol^i_{worst} et Sol^i_{best} est inférieure à une limite τ_{reset}, les phéromones sont réinitilisées à leur valeur initiale, et $\beta\%$ des chemins de Sol_{best}, sélectionnés aléatoirement, deviennent tabous jusqu'à la prochaine réinitilisation. Lorsqu'un chemin est tabou, plus aucune phéromone ne peut y être déposée.

4 Résultats

Pour tester notre algorithme, nous avons généré des instances [1], reprenant toutes les caractéristiques du problème de SimUSanté, à partir de celles du CB-CTT. Nous comparons les résultats obtenus par *SimUTACO*, à un modèle mathématique [2] implémenté sous CPLEX, ainsi qu'à *SimUVNS* [2], une recherche par voisinages variables. Le modèle sous CPLEX parvient à obtenir des résultats optimaux uniquement sur de petites et moyennes instances. *SimUVNS* et *SimUTACO* obtiennent respectivement des résultats ayant un écart moyen de 5.55% et 0.15% avec les résultats optimaux de CPLEX.

Références

[1] Simon Caillard, Laure Brisoux-Devendeville, and Corinne Lucet. Health Simulation Center Simusanté®'s Problem Benchmarks. https ://mis.u-picardie.fr/en/Benchmarks-GOC/.

[2] Simon Caillard, Laure Brisoux Devendeville, and Corinne Lucet. Variable neighborhood search for a planning problem with resource constraints in a health simulation center. *Applied Intelligence*, 09 2021.

[3] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In *Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, UK, August 29 - September 1, 1995, Selected Papers*, volume 1153 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 1995.

[4] Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The Second International Timetabling Competition (itc-2007) : Curriculum-Based Course Timetabling (track 3). Technical report, 2007.

[5] Marco Dorigo and Luca Maria Gambardella. Ant colony system : a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1) :53–66, 1997.